

<<软件测试从入门到精通>>

图书基本信息

书名：<<软件测试从入门到精通>>

13位ISBN编号：9787121113260

10位ISBN编号：7121113260

出版时间：2010-7

出版时间：电子工业

作者：王轶辰

页数：286

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

## <<软件测试从入门到精通>>

### 前言

笔者从事软件测试已近十年，从跟着导师编测试程序的研究生，到完成了关于软件测试研究论文的博士，再到目前成为一个软件测评中心的技术负责人，基本上走过了入门、摸索、提高的路程。在软件测试领域摸爬滚打了这些年以后，总想把自己的一些感触和领悟写出来，希望能够对想要加入这个行业或者已经在这个行业中的人们有些作用。

“从入门到精通”，这个思路很好，因为我们都是这样走过来的（虽然路很长，我们还在途中）

。但是这样的主题写起来并不容易，因为走过这个历程的人很少还能够记得自己入门时的情景了。

我在试图重新构建从入门到精通的这段路程时，使用了以下的方法： 1.从我们测评中心的档案库中找到了我为不同阶段做过的软件测试项目所编写的全部测试文档，以现在的眼光去重新审视。

2.对测评中心内不同职称的测试人员进行访谈，将结果用思维导图的形式表示出来，并进行分析

。 3.参考了一些市面上常用的测试教科书。

经过一番探讨，我整理出测试历程的三个阶段：入门期、提高期、精通期。

入门期 从入门期的思维导图中我们可以看出，一个不懂软件测试的人所想到的内容几乎能够涵盖软件测试的方方面面。

## <<软件测试从入门到精通>>

### 内容概要

随着软件应用越来越广泛，如何提高软件的质量和可靠性成为软件工作者必须应对的挑战。而软件本身具有“看不见摸不着”的特点，使对软件的验证和测试与对其他产品的验证和测试大相径庭。

本书从软件测试的基本概念讲起，循序渐进地为读者讲解软件生命周期各个测试阶段应该完成的任务和采用的方法。

书中涉及的项目实例多为作者及所在团队参与的课题，具有很强的指导和借鉴意义。希望读者能够从这本书中获取足够的软件测试知识，成为合格的软件测试工作者。

本书适合软件测试的初学者与具有一定测试经验的人员使用。

## &lt;&lt;软件测试从入门到精通&gt;&gt;

## 书籍目录

第1篇 入门篇 第1章 软件测试的基本概念 1 1.1 软件测试的定义 1 1.1.1 软件测试定义的发展 1 1.1.2 对软件测试的正确认识 2 1.1.3 软件测试概念的深入理解 9 1.1.4 软件测试定义的再讨论 13 1.2 软件测试的概念模型 14 1.2.1 测试目标 14 1.2.2 测试对象 20 1.2.3 测试依据 21 1.2.4 缺陷定义 21 1.2.5 测试解决方案 24 1.2.6 测试结果 25 1.3 软件测试的分类 25 1.3.1 测试目标的实例化 25 1.3.2 测试对象的实例化 26 1.3.3 测试依据的实例化 26 1.3.4 测试方案的实例化 26 第2章 软件测试的基本方法 28 2.1 审查技术概论 29 2.2 代码审查技术 30 2.2.1 几个基本概念 30 2.2.2 代码审查的依据 31 2.2.3 代码审查的要求 32 2.2.4 代码审查的结果 33 2.2.5 代码审查的文档 36 2.2.6 代码审查的策略 38 2.3 文档审查技术 41 2.3.1 目的与内容 41 2.3.2 文档审查的流程 43 2.3.3 文档审查的策略 44 2.4 自动化静态测试技术 47 2.4.1 基于模式(规则)的静态代码分析 47 2.4.2 程序的静态结构分析 49 2.4.3 代码度量计算 51 2.4.4 自动化静态测试技术的使用策略 52 2.4.5 使用自动化工具进行静态测试 53 2.5 白盒测试技术 54 2.5.1 逻辑覆盖测试 54 2.5.2 基本路径测试 57 2.5.3 循环结构测试 59 2.5.4 程序插桩测试 60 2.5.5 白盒测试方法的综合使用策略 60 2.6 黑盒测试技术 61 2.6.1 功能分解法 61 2.6.2 等价类划分法 61 2.6.3 边界值分析法 62 2.6.4 因果图方法 62 2.6.5 随机测试方法 66 2.6.6 猜错法 66 2.6.7 黑盒测试方法的综合使用策略 66 第3章 软件测试的框架表示 67 3.1 测试框架的概念 67 3.2 测试框架的表述 69 3.2.1 原则层 69 3.2.2 结构层 70 3.2.3 细节层 70 3.3 测试框架的特征与优势 70 3.4 测试框架的质量 71 3.5 基于测试框架的软件测试 72 3.5.1 基于测试框架的软件测试过程 72 3.5.2 测试框架的扩展 73 3.5.3 测试框架的实例化 74 3.6 测试框架的设计 74 3.6.1 设计原则 74 3.6.2 测试框架的设计方法 75 第2篇 提高篇 第4章 软件测试过程 77 4.1 软件的生命周期模型 77 4.1.1 瀑布模型 77 4.1.2 V模型 78 4.1.3 螺旋模型 79 4.1.4 统一的软件开发过程 80 4.2 软件开发与软件测试 84 4.2.1 软件测试的生存周期模型 84 4.2.2 软件测试的分级 84 4.2.3 全生命周期测试的基本原则 85 4.3 软件测试过程模型 86 4.3.1 测试策划 86 4.3.2 测试设计与实现 90 4.3.3 测试执行 93 4.3.4 测试总结 94 第5章 软件单元测试及实践 96 5.1 基本概念 96 5.2 单元测试的目标 96 5.2.1 单元测试的要求 96 5.2.2 单元测试的内容 96 5.3 单元测试的策略 98 5.3.1 静态与动态结合的测试 98 5.3.2 单元测试中的覆盖率 98 5.3.3 单元测试的自动化意义 100 5.3.4 单元测试与项目开发 100 5.3.5 单元测试中的功能测试 101 5.3.6 嵌入式软件单元测试 101 5.4 单元测试的过程 104 5.4.1 单元测试的计划 104 5.4.2 单元测试的设计 106 5.4.3 单元测试的执行 107 5.4.4 单元测试的结果分析 107 5.5 单元测试环境 108 5.5.1 单元测试自动化 108 5.5.2 单元测试工具概论 109 5.6 单元测试的实践 109 5.6.1 一段实例代码 110 5.6.2 单元测试策划 113 5.6.3 单元测试设计 115 5.6.4 单元测试报告 118 第6章 软件集成测试及实践 119 6.1 集成测试的要求与内容 119 6.1.1 集成测试的要求 119 6.1.2 集成测试的内容 119 6.2 集成测试的策略 121 6.2.1 基于分解的集成策略 121 6.2.2 分层式集成测试 123 6.2.3 集成的McCabe基本路径 124 6.2.4 调用流的集成 125 6.3 集成测试的过程 127 6.3.1 集成测试的计划 127 6.3.2 集成测试的设计 128 6.3.3 集成测试的执行 130 6.3.4 集成测试的结果分析 130 6.4 集成测试的实践 131 第7章 软件系统测试及实践 134 7.1 系统测试的基本概念 134 7.2 系统测试的要求 135 7.3 系统测试的策略 135 7.3.1 功能测试 135 7.3.2 性能测试 136 7.3.3 接口测试 136 7.3.4 人机交互界面测试 136 7.3.5 强度测试 137 7.3.6 安全性测试 138 7.3.7 余量测试 139 7.3.8 恢复性测试 139 7.3.9 安装性测试 140 7.3.10 边界测试 140 7.3.11 敏感性测试 140 7.3.12 互操作性测试 140 7.3.13 容量测试 141 7.3.14 数据处理测试 141 7.3.15 可靠性测试 141 7.4 系统测试的过程 141 7.4.1 系统测试的策划 141 7.4.2 系统测试的设计 144 7.4.3 系统测试的执行 145 7.5 系统测试环境 147 7.6 系统测试的实践 148 7.6.1 一个需求实例的介绍 148 7.6.2 系统测试需求的分析 148 7.6.3 系统测试设计 149 第8章 软件验收测试 152 8.1 验收测试的基本概念 152 8.2 验收测试的内容 152 8.3 验收测试的策略 153 8.3.1 验收测试的类型 153 8.3.2 验收测试的进入条件 153 8.3.3 网络软件的验收测试 153 8.3.4 软件验收测试的充分性 154 8.3.5 验收测试的执行 154 8.4 软件验收的过程 154 8.4.1 软件验收申请 154 8.4.2 制定软件验收计划 155 8.4.3 成立软件验收组织 155 8.4.4 软件验收测试和配置审核 156 8.4.5 软件验收评审 157 8.4.6 软件验收报告 158 8.4.7 软件产品交付 158 第9章 软件测试管理 159 9.1 测试项目管理概述 159 9.1.1 测试项目概述 159 9.1.2 测试项目管理 160 9.1.3 测试项目管理

## &lt;&lt;软件测试从入门到精通&gt;&gt;

的三维模型 162 9.2 软件测试的全过程管理 163 9.2.1 测试计划管理 163 9.2.2 测试设计与分析管理 170 9.2.3 测试执行过程管理 173 9.2.4 测试结果的管理 173 9.3 软件测试的全方位管理 174 9.3.1 软件缺陷的管理 174 9.3.2 回归测试的管理 176 9.3.3 测试文档的管理 176 9.3.4 测试评审的管理 177 9.3.5 测试的配置管理 181 9.3.6 测试质量的管理 182 9.4 测试人员的管理 184 9.4.1 测试人员的基本素质 184 9.4.2 测试小组的管理 185 9.4.3 测试组织的管理 187第3篇 精通篇 第10章 软件测试的抽象模型及数学描述 191 10.1 测试目标 191 10.2 测试对象 191 10.2.1 基本概念 191 10.2.2 数学描述 192 10.3 测试依据 194 10.3.1 基本概念 194 10.3.2 数学描述 195 10.4 缺陷定义 195 10.4.1 基本概念 195 10.4.2 数学描述 196 10.5 测试解决方案 197 10.5.1 基本概念 197 10.5.2 数学描述 198 10.6 测试结果 202 10.6.1 基本概念 202 10.6.2 数学描述 202 10.7 书中数学符号的说明 203 第11章 基于需求的软件测试 204 11.1 软件测试过程概述 204 11.2 测试环境 205 11.3 基于需求的测试用例的选择 205 11.3.1 正常范围测试用例 205 11.3.2 健壮测试用例 205 11.4 基于需求的测试方法 206 11.4.1 基于需求的硬件/软件综合测试 206 11.4.2 基于需求的软件综合测试 206 11.4.3 基于需求的低级测试 207 11.5 测试覆盖分析 207 11.5.1 基于需求的测试覆盖分析 208 11.5.2 结构覆盖分析 208 11.5.3 结构覆盖分析方法 208 11.6 MC/DC覆盖率 208 11.6.1 对C/DC和MC/DC的描述 208 11.6.2 C/DC和MC/DC之间的差异 209 第12章 嵌入式软件的仿真测试框架 212 12.1 仿真测试框架的提出 212 12.1.1 嵌入式软件的特点 212 12.1.2 嵌入式软件的测试 213 12.1.3 解决方案的抽象 214 12.1.4 仿真测试框架 214 12.2 仿真测试框架的测试域原则 215 12.3 仿真测试框架的基本原理 215 12.3.1 仿真测试框架的基本原理 215 12.3.2 仿真测试原理 216 12.3.3 模型驱动测试的原理 221 12.4 仿真测试框架的使用原则 226 12.5 测试框架体系结构描述 226 12.6 仿真测试框架的体系结构 227 12.6.1 测试组件视图 227 12.6.2 测试过程视图 228 12.6.3 测试组织管理视图 230 12.6.4 测试工具视图 231 12.6.5 测试文档视图 232 12.7 仿真测试框架的细节层 234 12.8 仿真测试框架的文档体系 234 第13章 典型工程应用 238 13.1 项目背景 238 13.2 仿真测试框架的实例化 238 13.3 基于仿真测试框架的测试 239 13.3.1 需求分析 239 13.3.2 测试设计 243 13.3.3 测试执行 248 13.3.4 测试分析 248附录A 软件测试常用术语 250附录B 系统测试需求规格说明模板 270附录C 系统测试计划模板 273附录D 系统测试说明 279附录E 系统测试报告模板 282参考文献 287

## 章节摘录

(2) 过平地开始规则检查会耽误编码的时间 很多软件开发人员抱怨过早地开始测试会占用他们宝贵的开发时间, 因此对于早期开展静态测试有很大的抵触情绪。

这是个思维上的误区。

有没有想过, 如果本来能够在早期通过规则检查就发现的缺陷, 一直带到系统测试阶段才通过故障现象被揭露, 其修改过程会更加耽误工程进度?

现在有些自动化的测试工具, 可以与版本控制软件, 如CVS或Sub.verSion无缝地集成, 成为软件开发过程中有机的组成部分, 不仅不会耽误开发时间, 反而能提高软件编制效率, 因为减少了返工修改错误的时间。

尽早开始测试, 会带来另一个负面的问题, 那就是代码是否在编写完成之后就立刻开始测试。

这一点, 从工程经验上看, 我们的建议是, 代码需通过编译后再开始规则检查。

一来可以避免由规则检查来完成本应该由编译器完成的语法检查工作, 二来编译的结果有助于规则检查之后的其他静态分析和测试过程。

(3) 编码规则妨碍了编程人员的灵感 软件工程师与艺术家有相同之处, 都是在创作一件作品。

只不过艺术家创造出来的是有形的艺术品, 而软件工程师创造出来的是无形的软件。

编写软件是一件创作型工作, 其中就难免带有编程人员的个人风格。

这就好像每个人的书法都有自己个性, 但是, 如果要写一本让大家都能读懂的作品, 我们还是喜欢使用印刷体, 惟其正规易懂。

如果每个编程人员都使用自己的风格编写代码, 最终的软件产品将难以保证统一的质量, 更不用说可维护性了。

严格遵守编码规范可能会在某种程度上妨碍编程人员设计华丽精巧的代码, 却能够大大提高最终软件产品的可维护性和可读性。

(4) 静态规则检查只要做一次就够了 静态分析中的另一个误区是, 有“一劳永逸”的办法, 即扫描整个代码库一次, 然后在诸如定型等重要节点前解决所有静态代码分析报告的问题。

在实际项目中, 有很多时候都是这样做的, 但是这样静态分析结果经常令人沮丧。

原因在于以下两方面: 发现的缺陷太多(两万行代码发现一万个缺陷)。

由于隐含缺陷太多, 时间又不允许详尽地排查, 以至于只好睁一只眼闭一只眼, 结果就是任何缺陷都不能被解决。

静态分析发现了严重的错误隐患, 是否修改代码成了难题。

修改代码就意味着已经完成的集成测试或系统测试必须重做, 而不修改代码在情理上说不通。

正如软件测试应该是伴随软件生命周期的工作一样, 静态分析也应该是伴随软件生命周期的测试策略。

这里还要强调一下: 静态分析是一种测试策略而不是测试阶段。

一定要避免“先做静态分析再做单元测试”的思维。

如果在系统交付的最后阶段, 出于应付差事而进行规则检查。

只能让开发过程愈发混乱, 修改代码难度加大。

## <<软件测试从入门到精通>>

### 编辑推荐

《软件测试从入门到精通》循序渐进地为读者讲解软件生命周期各个测试阶段应该完成的任务和使用的方法，使读者对软件测试有一个全面认识，了解各类软件测试的执行过程与方法，软件测试的自动化工具使用，并能够解决具体问题。

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>